ECSE 2920: Design Methodology

Deliverable 4

Group 11

01/29/2025

Part 1: Transfer Function

Requirements: Using optical encoders, calculate and solidify a transfer function that successfully makes both DC motors move at the same rate of speed. Include a one page write up for the transfer functions. How did you get it, what is it, how do you use it?

To obtain our transfer functions using the optical encoders we used our PI to drive the motors and used the PI to take input from the Optical encoders. We used a python program to write to a csv file for different RPM output for duty cycle input for both the left and right motor. We did 3 trials for each motor and then averaged the average RPM for duty cycles ranging from .65 to .99. We then used a different python program to plot the transfer functions for each motor using the duty cycle as the input and the average RPM as the output. We also used this same python program to fit this data using a logarithmic function. Using the 2 transfer functions obtained from our python program, we plugged both of them into a graphing software called Desmos to view the plots of both graphs. Because the left motor output less RPM per duty cycle, we gave the left motor a duty cycle multiplier until the graphs approximately overlapped. Using this multiplier to our left motor's duty cycle and staying in a certain duty cycle range we were able to get the motors to spin at the same rate of speed.

To get RPM from the optical encoders we used a while loop and incremented a counter variable every time the encoder detected a rising edge. This while loop ran for 10 seconds so we divided the amount of counter by 20 (there are 20 slots in the wheel, that's 20 rising edges per rotation) and then times by 6, to convert to minutes. This gave us RPM. We then wrote this value to a csv using the RPM as the output and the duty cycle that gave us the RPM as the input. After doing this for both wheels 3 times and averaging them. We plugged these averages into another python program in the form of arrays. We had four arrays, two for both motors, one for duty cycle and one for RPM. Using numpy, matplotlib, and scipy we were able to plot the points and obtain logarithmic functions for each motor. Then we went to Desmos and defined the left motor function as L(x) and the right motor function as R(x) and plotted them both. Then because the L(x) function was "weaker" we gave it a multiplier, so we graphed L(ax) and added a slider to adjust the value of a. We found that a = 1.088 made the 2 functions overlap at a duty cycle of .81. This means that if we spin the right motor with a duty cycle of .81 and the left motor with a duty cycle of .81 * 1.088 the motors will spin at the same RPM at around 54.5 RPM.

Picture(s)

```
while (time.time() - start) < 10:

    if (status == 0 and encoder.value == 1):
        count += 1

    status = encoder.value

Rotations = (count/20)
RPM = Rotations*6
motor.stop()
print("Motor: " + Motor + " | Duty Cycle: " + str(Duty) + " | RPM: " + str(RPM) )
```

(This is the segment of code that obtains the RPM)

```
with open('RPMs.csv', 'w', newline='') as csvfile:
    spamwriter = csv.writer(csvfile, delimiter=' ',
                            quotechar='|', quoting=csv.QUOTE_MINIMAL)
    spamwriter.writerow(['Motor: ' + Motor, 'Duty Cycle: ' + str(Duty), 'RPM: ' + str(RPM)])
```

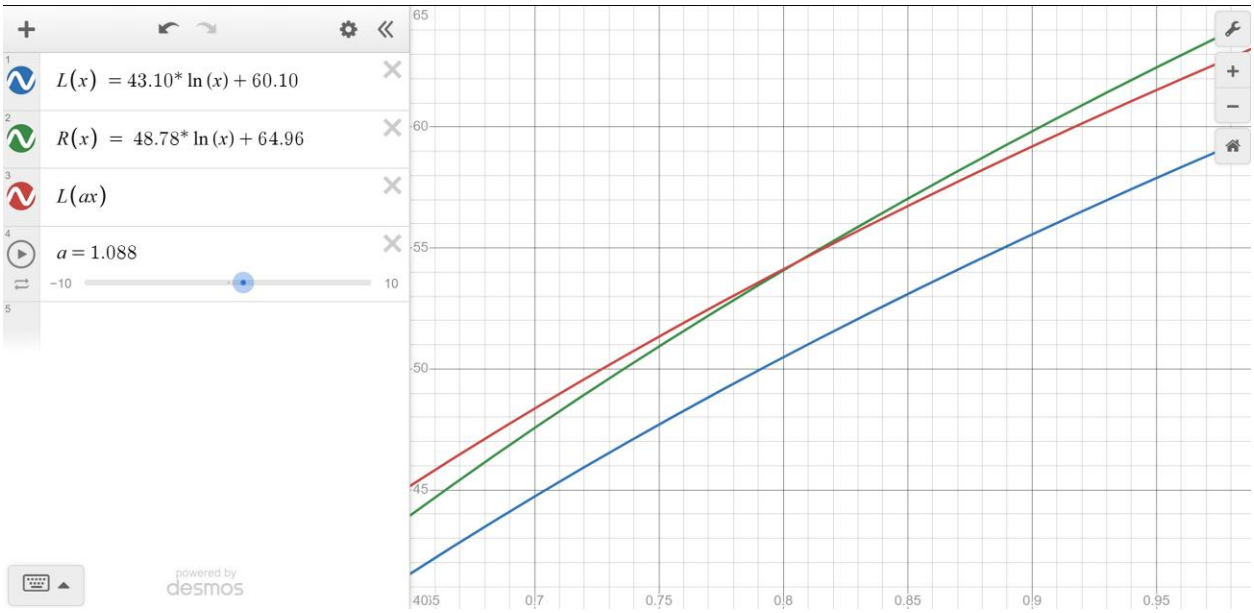
(This is the segment of code that wrote the RPM and duty values to a CSV)

```
# Define the logarithmic function: y = a * ln(x) + b
def log_func(x, a, b):
    return a * np.log(x) + b

# Fit both data sets to the logarithmic model
params1, _ = curve_fit(log_func, x1, y1)
params2, _ = curve_fit(log_func, x2, y2)

# Extract parameters for both fits
a1, b1 = params1
a2, b2 = params2
```

(Segment of code used to fit the data in the x1 y1 x2 y2 (duty cycle and RPM for left and right motor) arrays to a logarithmic function.)



(Desmos screen shot of the 2 functions and the multiplier)

Key Design Decision(s)

- What did your team clarify about the design?
    - There is a clear tradeoff between autonomy and hard coding in a hardware-based situation. As we make something more complex, we run into more issues and possibly understand less about how to find a solution, but if working correctly, will produce a better result than the simpler solution.
- What were the competing choices in the design?
    - Hard Coding a transfer function for a single duty cycle (Simplest)
    - Hard Coding a transfer function for various duty cycles
    - Creating an autonomous PID that corrects the driving in real-time based on optical encoder values (Most Complex)
- Ultimately what did your team choose and why?
    - As of now, we are trying the first and simplest option. We can always scale up if needed and not waste as much time as trying the most complex option and scrapping it. While we would like the most optimal solution for our car, we are also taking time into account, and feel that it is worth trying the simplest solution first.

TEST… Test… test

- What aspects of the design need to be tested?
    - Software?
        - We needed to test sample functions that drive the motors in specific actions. (Turn Left, Turn Right, Straight, Square)
        - We needed to test and modify the found adjustment values to maintain smooth driving
    - Hardware?
        - We moved to a new chassis to avoid the wheels scraping on the sides and moved the motor driver to a breadboard of a smaller form factor
        - One Optical Encoder needs to be tested in order to see if it is broken or is supplying output voltages below the GPIO threshold.
- Who is responsible for testing? Cade Toney, Michael See, Dawson Gulasa
    - Tests ran?
        - Motor Driver works after rewiring and wheels no longer scrape against chassis.
        - We ran the sample functions and found the car to consistently be veering to the left, implying that the right motor is still spinning faster than the left, despite the adjustment multiplier.
        - The Optical encoder supplies a voltage lower than the GPIO threshold. Amplification is required (Op-Amp or Transistor)
    - Conclusions from testing?
        - We will have to move to a more complicated solution than what is present, because the problem is more complicated. The motors cannot drive straight just from linear adjustment and will need to take advantage of the encoders in real time to steer correctly.

summary/part conclusion (make sure you address all parts of the requirements)

In conclusion we used the optical encoders, the PI, and python to make 1 transfer function for each motor. We then graphed these transfer functions using a graphing tool and manipulated the weaker transfer function with a multiplier that can be applied to the duty cycle to make them spin the same speed. Although this didn't completely solve the steering issue, it is a step in the right direction, and we now recognize that a more robust solution is necessary for securing success in the big picture of this project.

Part 2: GPIO and Power Diagrams

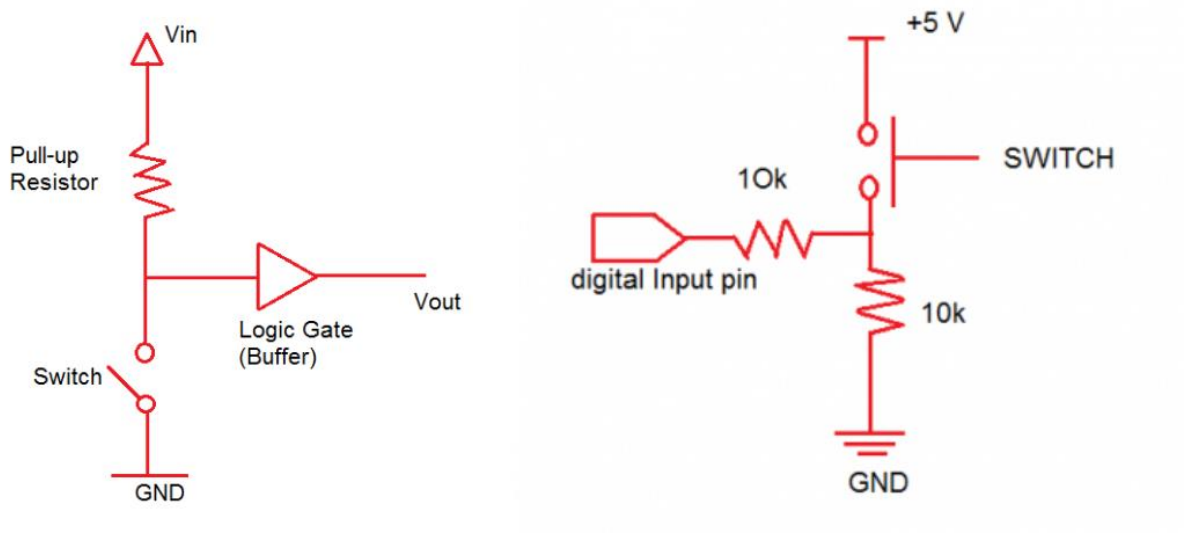
Requirements: Draw and describe the power diagram and program GPIOs

GPIO Diagram

GPIO names of PI	Functional name for project	Default state at power up	State used for project	Pull up or pull down (if any)	Switch needed to set GPIO
12	Forward pin for left motor	low	High When left motor spins forward, low when left motor spins backward	Pull down	N/A
13	Backward pin for left motor	low	High when left motor spins backward, low when left motor spins forward	Pull down	N/A
18	Forward pin for right motor	low	High when right motor spins forward, low when right motor spins backward	Pull down	N/A
19	Backward pin for right motor	low	High when right motor spins backward, low when right motor spins forward	Pull down	N/A

Explanation of pull-up and pull-down resistors and how they can be used, relate to your audio car:

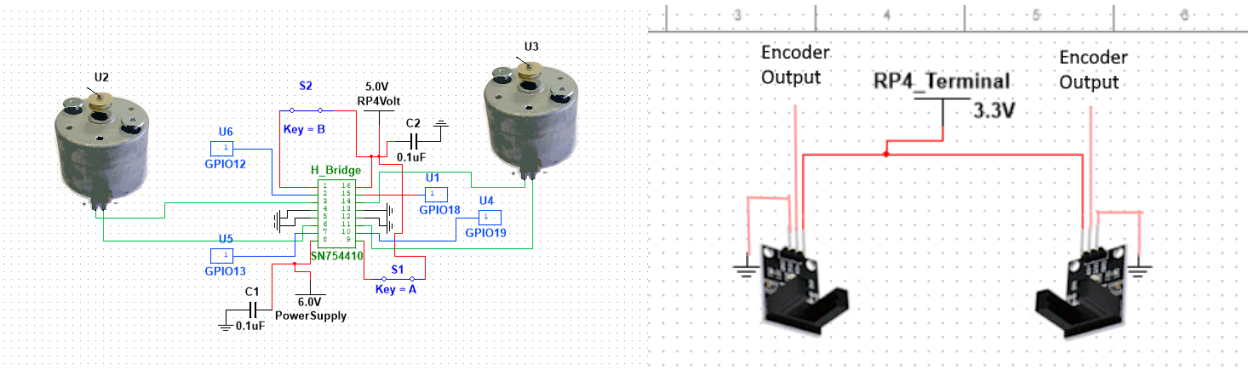
Pull up and pull down resistors are used to eliminate floating voltage values. For example when using a square wave on a pin that defaults low there is a switch to ground. When the pin is low the switch is closed and the signal will be low. However when the signal wants to go high it will disconnect the switch and it will have a floating voltage. To fix this you use a pull up resistor. This is a resistor placed between your signal and V+ when the switch is closed there is no current flowing through the resistor (no change in voltage) and the signal will be high. A pull down resistor works very similarly. When a pin defaults high there is a switch between signal and V+. When the switch is closed the signal is high; however, when the switch is open the signal will have a floating voltage value. You can fix this with a pull down resistor. You connect the signal to ground through a resistor and when the switch is open there is no current through the resistor and ground and signal will have the same voltage, therefore fixing the floating voltage issue. Pull up and pull down can be used in relation to the audio car when using the optical encoder to measure RPM. The optical encoder senses light and sends rising and falling edges to the PI. If you're using pins that default low you would want to use a pull up resistor to pull the signal up to V+ when a rising edge is sensed, instead of be a floating voltage and not being an accurate reading.



Power Diagram

Name of the Device	Pins that require power (voltage requirement and current requirement)	Name of source delivering the required power (voltage and current)
Left Motor	Pin 3	5 Volt RP4 terminal
Right Motor	Pin 14	5 Volt RP4 terminal
Optical Encoder for left Motor	Connected Directly from 3.3 Volt terminal to optical encoder	3.3 Volt RP4 terminal
Optical Encoder for right motor	Connected Directly from 3.3 Volt terminal to optical encoder	3.3 Volt RP4 terminal

Picture(s)



(Circuits showing how our components receive power)

Key Design Decision(s)

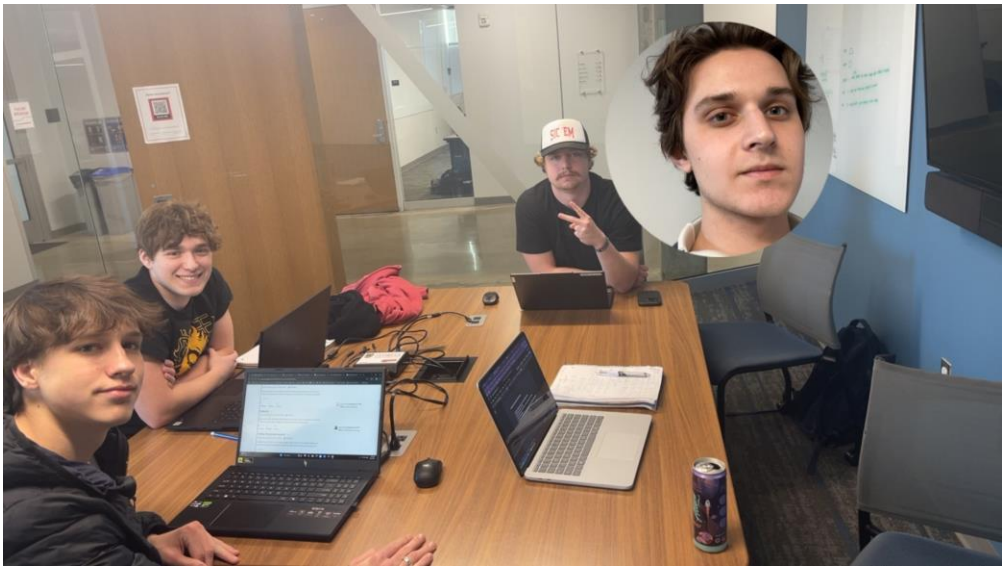
Since deliverable 3 the only additional component we have included that requires power is our optical encoders, so a majority of the key design decisions will be like last week's, here are additional remarks surrounding the encoders:

- What did your team clarify about the design?
    - GPIO configuration: determined the GPIO pins controlling motor direction would default to low at power-up and should use pull-down resistors to avoid floating voltage states
    - We clarified that the motor control pins need pull-down resistors, while the optical encoders require pull-up resistors to maintain signal integrity
    - We verified that no manual switches were needed to control the GPIOs, as the motor H-bridge is controlled programmatically. However, external switches could be used for additional manual control if necessary.
- What were the competing choices in the design?
    - We considered whether to use pull-up or pull-down resistors on the motor control pins.
    - There was an option to integrate a manual switch to override GPIO motor control, but we opted against it since the system is fully software-controlled
    - We explored different ways to process encoder output, including software debouncing v.s. hardware filtering.
- Ultimately what did your team choose and why?
    - We will permanently have the encoders attached to our audio car. This is very important in the long run so we can track the RPM of both of our motors and develop code surrounding the data so that we can get our sound car to move as we wish.
    - Pull-down resistors for Motor GPIOs to ensure a default low state at power-up preventing unexpected motor activation
    - No manual switches

summary/part conclusion (make sure you address all parts of the requirements)
Our design ensures stable power distribution and reliable GPIO programming for motor control and encoder feedback. The power diagram provides 5V from the RP4 terminal for the motors and 3.3V for the optical encoders, ensuring proper voltage levels for all components. To prevent floating voltages, we implemented pull-down resistors on motor control GPIOs. GPIOs were programmed to control motor direction via an H-bridge circuit, enabling precise movement.

Conclusion and Participation (REQUIRED)

1. **Include a selfie/photo from your group meeting/zoom call.**



2. **When did you meet?** 1/28/2025
3. **Who was present?** Dawson Gulasa, Finn Morton, Michael See, Cade Toney
4. **Who was not present?** Nobody was absent
5. **What were the main ideas discussed or major decisions (1-2 sentences/bullet points)**
    - We discussed and decided to move forward with the autonomous driving solution and updated each other on modern procedures when it came to programming practices and ssh startup/connection. We also finished up this document.